基于相邻层间相似性和空体素跳跃的体绘制加速算法研究 *

敖　山，刘梦颖，李保锟，刘志中

(河南理工大学 计算机科学与技术学院, 河南 焦作 454000)

摘　要：Splatting 是经典的基于物序的直接体绘制方法，运算数据量的多少制约着算法绘制图像的速度。为了进一步提升绘制速度，采用基于相邻层间相似性和空体素跳跃相结合的方法进行加速，在读取数据过程中对图片中的三维纹理数据进行筛选，并使用足迹表对筛选后的三维纹理数据进行二维投影，利用相邻层间相似性计算每一个点的灰度值，并根据灰度值将数据分类，算出对成像没有影响的空体素，跳过其绘制过程从而加速算法。实验结果显示：提出算法能够在保证绘制图像质量的基础上，在一定程度上解决和改善了 Splatting 算法数据的空间相关性和运算效率的问题。

关键词：体绘制；Splatting 算法；足迹表；相邻层间相似性；空体素

中图分类号：TP391.4　　doi: 10.19734/j.issn.1001-3695.2019.12.0704

Research on accelerating volume rendering algorithm based on similarity between adjacent-layers and void voxel jump

Ao Shan, Liu Mengying, Li Baokun, Liu Zhizhong

(*School of Computer Science & Technology*, *Henan Polytechnic University*, *Jiaozuo* 454000, *China*)

Abstract: Splatting is a classical direct volume rendering method based on object order, which volume data exists in layers and each layer of data has similar lines. The amount of calculation data restricts the speed of image rendering. In order to further improve the rendering speed, this paper used a method based on the combination of similarity between adjacent layers and empty voxel jump to speed up the algorithm. It filtered the 3D texture data of the image in the process of reading the data, and then used the footstep table in the 3D texture data after filtering was projected in two dimensions. It calculated the gray value of each point by using the similarity between adjacent layers, and classified the data according to the gray value of each point to calculate the empty voxel that had no effect on the imaging, skiped the rendering process and speeded up the algorithm. The experimental results show that the optimized algorithm can solve and improve the spatial correlation and operation efficiency of splatting algorithm to a certain extent on the basis of ensuring the quality of the drawn image.

Key words: volume rendering; splatting algorithm; footprint method; comparability of adjacency-layers; void voxel

0　引言

体绘制[1]被普遍应用于医学 CT 成像、气象实时观测、地质勘测、分子生物学扫描成像、宇航等领域，在不同领域的应用对应着不同侧重的体绘制算法，如基于图像空间的光线投射算法(ray casting)[2]、基于物体空间的溅射算法(Splatting)和错切—变形算法(ahear-warp)[3]、基于频域的体绘制(frequency domain volume rendering)[4]等。经过多年的发展，如今的体绘制研究领域主要针对两大方向：如何提高绘制的图像质量和如何降低算法的绘制时间。通常绘制质量与绘制时间成正相关，如何在保证图像质量的基础上尽量降低算法绘制时间便是目前体绘制技术研究的热点之一。在众多体绘制算法中，光线投射算法计算得到的图片质量最高，但其算法复杂度相较于其他体绘制算法也更高，绘制时间更长。而 Splatting 算法绘制图像的质量虽然没有光线投射算法质量高，但是绘制时间相对较短。

随着计算机技术的发展，为了提高可视化的渲染速度，人们提出了许多体绘制加速算法，主要分为硬件和软件两大方向。基于硬件的加速主要取决于计算机硬件的发展，图形处理器的出现便为解决实时三维可视化问题提供了硬件支持，利用图像处理器的可编程性加速体绘制算法已然成为主流研究方向[5~7]，但从体绘制算法本身原理来看，上述方法并没有对之进行改善。基于软件加速的方法目前常用的有两种，一是通过对函数优化进行加速，如基于小波变换的多分辨[8]、基于提前不透明度截止[9]、基于并行和分布式体绘制[10]、基于密度-距离图的交互式分类[11]、结合图形硬件加速等 Splatting 算法，但是这些方法没有对数据进行过滤，在加速上并不完全。二是通过对数据处理提升绘制速度，如基于相邻层间相似性[12]，基于包围盒空间[13]，基于八叉树[14]、k-d 树[15]等数据结构方法，但是这些方法都没有加速算法过程，加速时间上还有很大的提升空间。

在相关研究的基础上，同时对函数改变和数据处理进行优化，通过研究运用 Splatting 算法进行体绘制时剔除无效数据和空体素跳跃，使绘制时间得到加速。设计对数据灰度值的过滤方法，在保证图像质量的同时，显著提升算法加速效果。

1　经典 Splatting 算法

Splatting 是一种基于空间数据扫描的经典体绘制算法，通过足迹函数逐层、逐行、逐个地计算每一个数据点对屏幕

收稿日期：2019-12-01；**修回日期：**2020-02-16　**基金项目：**国家自然科学基金资助项目(61872126)；全国教育科学规划教育部重点课题项目(DFA170292)；河南省软科学研究计划资助项目(182400410147)

作者简介：敖山(1971-)，男，四川丰都人，副教授，硕导，博士(后)，主要研究方向为计算机仿真和系统建模(drao@163.com)；刘梦颖(1993-)，女，河南开封人，硕士研究生，主要研究方向为计算机仿真和系统建模；李保锟(1997-)，男，河南获嘉人，学士，主要研究方向为图像处理；刘志中(1981-)，男，河南周口人，副教授，博士(后)，主要研究方向为服务计算、智能服务．

像素的贡献，利用算法投影到二维平面并加以合成，形成最终的图像[16,17]。Splatting 算法先将体数据表示为一个有交叠的基本函数构成的矩阵，继而通过将每一体素代入到足迹函数来计算体素投影的影响范围[18]，即图片进行重构的过程。用圆锥函数定义强度分布，计算出每一个体素对图像的总体贡献，从而将输入的图像数据转换到图像空间，再通过查找足迹函数提前计算得到的通用足迹表匹配每一个体素对于图像像素的影响值，并加以合成形成最终的图像。

在经典的 Splatting 算法流程中，重构是根据某个重构核，利用卷积运算将离散的三维数据场重构成连续数据的过程[19]，其数学表达式为

$$signal_{3D} = \iiint hv(u-x, v-y, w-z) \cdot \sum \delta(x,y,z)\rho(x,y,z)\,dudydw \tag{1}$$

其中：h_v 为重构核；u,v,w 是重构核的坐标；ρ 为三维数据场的密度函数；δ 为用于采样的梳状函数。

传统的体绘制算法在加载算法的时候，没有对最初的数据进行筛选，按层逐个对采样点进行绘制，使得查足迹表、计算每个体素所占屏幕像素的贡献值等的过程极其耗费计算能力和时间。针对这两点，研究通过相邻层间相似性和空体素跳跃的方法来加快运算速度，分别从相邻层间相似性、空体素跳跃法以及将两种改进方法相结合的三个方面来讨论 Splatting 算法优化设计。

2　体绘制加速算法优化

2.1　相邻层间相似性

在 Splatting 算法的卷积公式中，三重积分运算在进行大数据量数据场的重构过程中很难完成，但通过式(2)也可以完成对三维数据场的重构。

$$signal3D(x,y,z) = \sum_{i \in v} hv(i_x - x, i_y - y, i_z - z)\rho(i) \tag{2}$$

其中，i 是以 (x,y,z) 为中心的重构核内的采样点。

只考虑一个采样点对图像形成空间中多个点的影响，即求体素 $\langle i \rangle$ 对屏幕上的点 (x,y) 的贡献时，可以看做是体素 $\langle i \rangle$ 对空间上 (x,y,z) 点的贡献沿着 z 轴作积分，即

$$effect_i(x,y) = \int_{-\infty}^{\infty} hv(i_x - x, i_y - y, w)\rho(i)\,dw \tag{3}$$

观察式(3)可知，$\rho(i)$ 可以提出至积分外面，其大小便只与重构核及点 (x,y) 到重构核中心点的距离有关，因此，足迹函数可以定义为

$$\text{footprint}(x,y) = \int_{-\infty}^{\infty} h_v(x,y,w)\,dw \tag{4}$$

其中 (x,y) 是足迹函数中的像素点与重构核中心在图像上的映射点之间的距离。

在计算一个体素对图像的影响时，可以通过迅速查找在数据预处理阶段所计算出的表中所储存的足迹函数来进行加权计算，以求出这个体素对图像的影响。

通过观察可知，足迹函数只受所选的重构核影响。在足迹函数确定的情况下，便可以计算出位于足迹函数所覆盖区域内某一像素点的灰度值，即：

$$r(i,j,k), 1 \le i,j \le M, 1 \le k \le K \tag{5}$$

该式中，$r(i,j,k), 1 \le i,j \le M, 1 \le k \le K$ 是采样点在图像平面上的投影的点坐标。

文献[10]在采样三维体数据时，一般来说间隔在 1mm 左右，相对来说较小，以此来判断相邻层间体数据之间的相似性。设有三维体数据离散采样集 $r(i,j,k), 1 \le i,j \le M, 1 \le k \le K$，这里假设在 x,y 方向分辨率相同，符合大多数情况。两相邻层 L_{k1}，L_{k2} 在点 (i,j) 的密度(能量)值分别为: $r(i,j,k_1)$，$r(i,j,k_2)$，$1 \le i,j \le M$，根据式(5)对屏幕像素的贡献分别为

$$effect_{(i,j,k1)}(x,y) = r(i,j,k1)\int h_v(i_x - x, i_y, w)\,dw \tag{6}$$

$$effect_{(i,j,k2)}(x,y) = r(i,j,k2)\int h_v(i_x - x, i_y, w)\,dw \tag{7}$$

其变化为

$$\Delta effect(x,y) = (r(i,j,k2) - r(i,j,k2))\int h_v(i_x - x, i_y, w)\,dw \tag{8}$$

若两个相邻层 L_{k1} 和 L_{k2} 之间，点 (i,j) 处密度值没有改变，即对屏幕像素的计算贡献没有变化；如果密度值发生改变，对屏幕像素的贡献相应变化，由式(8)可知，其变化的大小相当于密度值为 $\Delta r = r(i,j,k2) - r(i,j,k1)$ 的点对屏幕像素的贡献。假设已经计算出了 L_{k1} 层所有数据点对屏幕像素的贡献，存储在 $effect_{(i,j,k1)}(x,y); 1 \le i,j \le M$ 中，这里不妨假设屏幕图像的分辨率仍为 $M*M$ 。计算 L_{k2} 层所有数据点对屏幕像素的贡献 $effect_{(i,j,k2)}(x,y); 1 \le i,j \le M$ 时，如果 L_{k2} 层的点 (i,j) 处密度值相比 L_{k1} 层点 (i,j) 处密度值没有改变，则所有的 $effect_{(i,j,k2)}(x,y)$ 不变，如果 L_{k2} 层的点 (i,j) 处密度值相比 L_{k1} 层点 (i,j) 处密度值发生 Δr 改变，仅需修改 $effect_{(i,j,k2)}(x,y)$ 中在该点影响范围(由足迹表决定)内的数据，具体修改值为 $P(y) = \sum_{i=1}^{n} p_i p(Y|i)$ 与足迹表相应权值之积。

2.2　空体素跳跃法

空体素是指通过 Splatting 算法重构，体素数据表现为透明状态，对最终图像的形成没有贡献的体素，例如灰尘等对真实图像合成没有贡献的要素，在绘制过程中将这些空体素跳跃过去可以提升算法的速度，并且不会损失绘制图像的质量。

在一个图像中有很多元素并存，每一个体素通常是由多种元素构成的，通过分析体素中不同元素存在的比例来判断哪些体素是不需要绘制的空体素。在一幅绘制的图像中，任意一个元素的灰度值 Y 的概率可表示为

$$P(y) = \sum_{i=1}^{n} p_i p(Y|i) \tag{9}$$

其中：n 为该体素所含有的元素种类的个数，p_i 为该体素中第 i 类元素的比例，$p(Y|i)$ 为第 i 类元素灰度值 Y 的条件概率。在了解每一种元素的灰度值概率密度分布函数的情况下，通过贝叶斯公式估算灰度值 Y 的体素中第 i 类元素的概率，即

$$P(i|Y) = \frac{p(Y|i)}{\sum_{j=1}^{n} p(Y|j)} \tag{10}$$

根据足迹表法计算得到体数据三维纹理后的灰度值，利用概率对采样点进行分类，空体素分类阈值为

$$key = \frac{p(y|i)}{\sum_{i=1}^{n} p(y|j)} \times 255 \tag{11}$$

如果每一个元素所对应的灰度值概率密度分布函数是未知的，通过对元素的灰度分布的原始曲线进行分析，求出其条件概率密度分布函数，再将灰度值和空体素分类阈值进行对比。如果判定灰度值大于空体素分类阈值，即进行光照和颜色合成操作，否则跳过，处理下一个采样点。

通过对元素进行分类，筛选需要绘制的元素。由式(11)可以得到空体素跳跃的公式为

$$Color = \begin{cases} \sum_{i=1}^{n} C_i p_i & value \ge key \\ 0 & value < key \end{cases} \tag{12}$$

空体素跳跃法通过跳过对空体素的灰度值计算和图像的颜色合成，减少对代码纹理存储器的访问次数，从而加快算法运算时间。空体素跳跃算法绘制出的图像，图像质量与原图像相比没有损失图像质量。

2.3　基于相邻层间相似性和空体素跳跃的 Splatting 算法

相邻层间相似性的加速算法利用体数据相邻层间差异较小的特性加快了计算采样点贡献值的过程，而空体素跳跃法是通过比较采样点的灰度值与空体素阈值，确定是否跳过该点之后的计算过程，以此提高图形绘制时间。两者结合可以在不损坏图像质量的基础上，对算法绘制时间进一步的加速，新的加速算法流程图见图 1。

a) 建足迹表。设定初始重构核大小，通过重构核建立通

用足迹表，在数据场计算重构核的空间卷积域时，使用给定的三维数据场的投影方向，再将体素与旋转核比例变换投影至二维平面，进一步利用足迹表求出足迹函数中平面投影与所覆盖像素点的数值实现三维卷积。

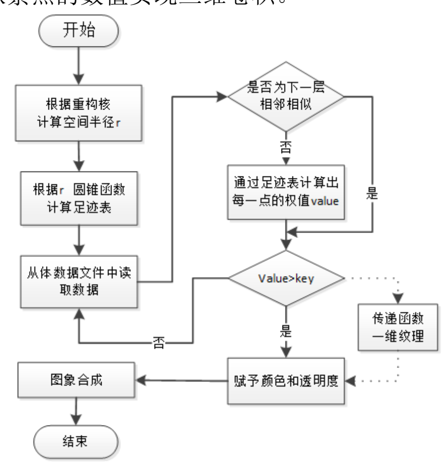


图 1　算法流程

Fig. 1　Algorithm flow chart

b) 从体数据文件读取数据过程中，对三维纹理数据进行筛选，当三维纹理数据 $(x,y,z)=0$ 时，将其视为无效数据剔除。

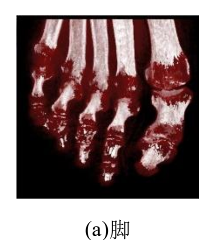
c) 通过用足迹表对筛选的三维纹理数据进行二维投影，利用相邻层间差异较小的特性计算每一个点的灰度值。若与下一层同样平面位置的点的差值为 0，则灰度值不变，若差值不为 0，则根据变化量计算该点的灰度值。

d) 通过灰度值将数据分类，跳过对图像没有贡献的空体素的绘制过程，将对图像有用的数据进行绘制，从而快速绘制出图像。

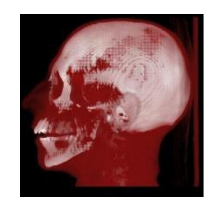
3　实验验证

3.1　实验环境与数据构造

通过加速算法参考时间为评价指标，对基于相邻层间相似性和空体素跳跃的 Splatting 算法进行评估。实验采用计算机配置为 Intel® Core™ i5-4210H CPU @ 2.90GHz，NVIDIA GeForce GTX 950M，win1064 位，12G 内存，在基于 OpenGL 的 C++环境下完成。实验所采用的是 CT 扫描中图像感应器，将捕捉到的光源信号转换为数字信号的原始数据。将包括脚，头正，头侧的一组图像作为实验对象(如图 2)。



(a)脚



(b)头(侧)



(c)头(正)

图 2　基于加速算法的 CT 扫描图像示例

Fig. 2　Example of CT scan image based on acceleration algorithm

3.2　测评结果与分析

1)基于相邻层间相似性

为了验证相邻层间相似性加速算法效果，设计 CT 图像成像速度的实验，分别对 Splatting 体素算法和基于相邻层间相似性算法处理后的数据分别进行绘制，并且对实验结果进行了比较和分析，得到两种算法在显示 CT 扫描三个实体的时间对比，如表 1 所示。

由测试结果可知，通过相邻体素的筛选可以有效的提高算法的速度，而且根据实验数据，脚图像加速到原图像耗费时间的 66.2%，头部图像加速到原图像耗费时间的 63%，圆形物体加速到原图像耗费时间的 78%，其均值为原 Splatting 算法的 69.1%。物体占图片的比例越低，其花费的时间越少，并且由于去除的都是无效的数据，对图像质量并没有影响。

表 1　基于相邻相似性加速算法的实验结果

Tab. 1　Experimental results based on adjacent similarity

绘制对象	绘制时间/s		加速比例
	Splatting 算法	基于相邻层间相似性	
脚	8.219000	5.440000	66.2%
头(侧)	9.641000	6.080000	63%
头(正)	8.406000	6.560000	78%
均值	8.755333	6.026667	69.1%

2)基于空体素跳跃算法

用空体素跳跃法绘制出的图像质量与原图像相比没有损失，能很好保证绘制图像的质量。基于改进的空体素跳跃算法进行图像的绘制实验，在加速效果上与原始 Splatting 绘制图像方法进行对比，如表 2 所示。

表 2　基于空体素跳跃加速算法的实验结果

Tab. 2　Experimental results based on void voxel skipping

绘制对象	绘制时间/s		算法加速比率
	Splatting 算法	空体素跳跃法	
脚	8.219000	3.79000	46.1%
头(侧)	9.641000	4.44000	46.1%
头(正)	8.406000	5.09000	60.6%
均值	8.755333	4.44000	50.9%

通过实验结果可以看出空体素跳跃法能够较大幅度的对算法进行加速，其耗费时间均值为原 Splatting 算法的 50.9%。并且随着式(10)算出的空体素分类阈值越发精确，以及需要绘制的元素数量越发的减少，使算法变得更加快速。在成像质量方面，由于跳跃的都是对成像没有贡献的体素，所以图像质量并没有改变。

3)基于相邻层间相似性和空体素跳跃的加速算法

基于相邻层间相似性和空体素跳跃的 Splatting 加速算法各有优缺点，把这两种处理体数据方法结合后，显示的各个实体的 CT 图像效果如表 3 所示，可以看出各个算法优化效果差别显著，其中脚图像加速到原图像耗费时间的 38.3%，头部(侧)图像加速到原图像耗费时间的 44%，头部(正)圆形物体加速到原图像耗费时间的 58.1%，其耗费时间均值为原 Splatting 算法的 46.8%。可以看出，基于相邻层间相似性和空体素跳跃相结合的 Splatting 算法绘制效果最为明显(图 3)，算法是高效可行的，并且由于两种算法的使用阶段不同，因此优化算法中两种加速算法能够相辅相成，达到更好的效果。

表 3　优化后的算法与 Splatting 算法的对比

Tab. 3　Comparison between the optimizedand Splatting algorithm

绘制对象	绘制时间/s		算法加速比率
	Splatting 算法	优化后的算法	
脚	8.219000	3.150000	38.3%
头(侧)	9.641000	4.240000	44%
头(正)	8.406000	4.887000	58.1%
均值	8.755333	4.092333	46.8%

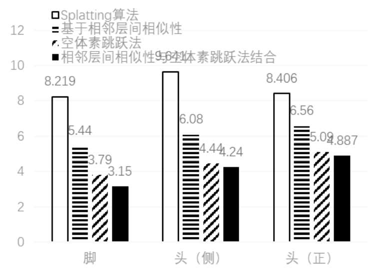


图 3 算法结果比较

Fig. 3　Comparison of results of algorithms

4 结束语

针对 Splatting 的加速算法的优化研究，在发挥原算法优势的基础上，实现了在不同阶段相邻层间相似性和空体素跳跃法相结合的加速算法，并对其进行了实验研究。结果表明优化后的算法能够在保证绘制图像质量的基础上，实现了绘制图像明显的加速效果。

虽然优化后的算法在一定程度上解决和改善了 Splatting 算法数据的空间相关性和运算效率的问题，但在图像绘制质量上与光线投射算法还有差距，还不能够明显改善 Splatting 算法所固有的色彩扩散问题，今后研究可以在分阶段加速算法中融入提升图片质量的思想，获得更好的效果。

参考文献：

[1] M. Levoy. Display of surfaces from volume data. IEEE Computer. Graph Application, 8 (3): 29-37, 1988.

[2] Zeng Yanyang, Pei Qingqing, Li Baokun. Ray-casting algorithm based on adaptive compound interpolation [J]. Journal of System Simulation, 2018, 30 (11): 4187-4194.

[3] Marathe C V, Gadre V M. Wavelet regularization for frequency domain volume rendering [C]. IEEE Communications, 2013.

[4] Piccand S, Noumeir R, Paquette E. Region of interest and multiresolution for volume rendering [J]. Information Technology in Biomedicine, 2008, 12 (5): 561-568.

[5] 朱奭，常晋义. 一种改进的基于 CUDA 的纹理映射和光线投射结合的体绘制算法 [J]. 计算机应用研究, 2015, 32 (06): 1884-1887. (Zhu Shi, Chang Jinyi, Improved algorithm of volume rendering combined texture mapping with ray casting based on CUDA [J]. Application Research of Computer, 2015, 32 (06): 1884-1887.)

[6] 何拥军，曾文权，余爱民. 基于 GPU 的三维医学图像体绘制技术综述 [J]. 计算机与数字工程, 2016, 44 (01): 141-147. (He Yongjun, Zeng Wenquan, Yu Aimin. Certain body of 3D medical image based GPU rendering technology summary [J]. Computer&Digital Engineering, 2016, 44 (01): 141-147.)

[7] 李林，鲁才，唐志梁，等. 基于数据流聚类策略的 GPU 码书初始化算法 [J]. 计算机应用研究, 2017, 34 (02): 426-430. (Li Lin, Lu Cai, Tang Zhiliang, *et al.* Codebook initialization algorithm based on data stream clustering using GPU [J]. Application Research of Computer, 2017, 34 (02): 426-430.)

[8] Hassan A H, Fluke C J, Barnes D G. A Distributed GPU-based Framework for real-time 3D Volume Rendering of Large Astronomical Data Cubes [J]. Publications of the Astronomical Society of Australia, 2012, 29 (3): 340-351.

[9] Beyer J, Hadwiger M, Pfister H. State-of-the-Art in GPU-Based Large-Scale Volume Visualization [M]. The Eurographs Association & John Wiley & Sons, Ltd. 2015.

[10] Cao Yi, Ai Zhiwei, Wang Huawei. A distributed multi-node GPU accelerated parallel rendering scheme for visualization cluster environment [C]. International Conference on Virtual Reality and Visualization. IEEE Computer Society, 2013: 153-160.

[11] 周芳芳，高飞，刘勇刚，等. 基于密度-距离图的交互式体数据分类方法 [J]. 软件学报, 2016, 27 (05): 1061-1073. (Zhou Fangfang, Gao Fei, Liu Yonggang, *et al.* Interactive Volume Data Classification Based on Density-Distance Graph [J]. Journal of Software, 2016, 27 (05): 1061-1073.)

[12] 张建勋，孙济洲，韩逢庆，等. 基于相邻层间相似性的加速 Splatting 算法 [J]. 系统仿真学报, 2005 (02): 421-423, 428. (Zhang Jianxun, Sun Jizhou, Han Fengqing, *et al.* The Accelerating Splatting Algorithm Based on Comparability of Adjacency-Layers [J]. Journal of System Simulation, 2005 (02): 421-423, 428.)

[13] 王睿，陈春晓，刘高，等. 基于自适应包围盒划分的体绘制加速方法研究 [J]. 仪器仪表学报, 2014, 35 (11): 2560-2566. (Wang Rui, Chen Chunxiao, Liu Gao, *et al.* Study on accelerated volume rendering method based on adaptive bounding box division [J]. Chinese Journal of Scientific Instrument, 2014, 35 (11): 2560-2566.)

[14] 李国和，段忠祥，吴卫江，等. 针对全空子数据体的 GPU 体绘制 [J]. 中国图像图形学报, 2014, 19 (04): 577-582. (Li Guohe, Duan Zhongxiang, Wu Weijiang, *et al.* GPU-based volume rendering for full-empty subdata blocks [J]. Journal of Image and Graphics. 2014, 19 (04): 577-582.)

[15] Zellmann Stefan, Schulze Jurgen P, Lang Ulrich. Binned k-d tree construction for sparse volume data on multi-core and GPU systems [J]. IEEE transactions on visualization and computer graphics. 2019 (03): 1941-0506.

[16] 李泽宇，陈一民，赵艳，等. 基于改进光线投影算法的医学图像三维重建 [J]. 计算机应用研究, 2017, 34 (12): 3866-3868, 3888. (Li Zeyu, Chen Yimin, Zhao Yan, *et al.* Three dimensional reconstruction of medical images based on improved ray projection algorithm [J]. Application Research of Computers. 2017, 34 (12): 3866-3868, 3888.)

[17] 贺楠楠. 医学图像三维重建算法研究 [D]. 河南：河南工业大学, 2018. (He Nannan. 3D reconstruction algorithm for medical images [D]. Henan: Henan University of Technology. 2018.)

[18] 田亮. 基于 CUDA 的足迹表法可视化研究 [D]. 南京：南京理工大学, 2011. (Tian Liang. Research on visualization of footprint method based on CUDA [D]. Nanjing: Nanjing University of Science and Technology, 2011.)

[19] 林金花. 基于空间体素融合的三维重建算法研究 [D]. 长春：中国科学院长春光学精密机械与物理研究所, 2017. (Lin Jinhua. Research on 3D reconstruction algorithm based on spacial voxel fusion [D]. Changchun: Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, 2017)